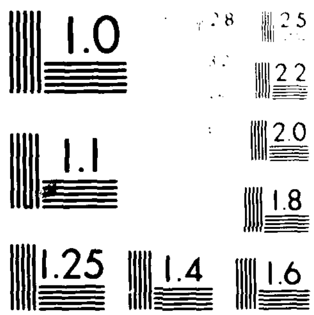
AD-A108 364 NAVAL RESEARCH LAB WASHINGTON DC F/G 5/8
ON DIGITAL PROCESSING VIA PDQ FACTORIZATION.(U)
NOV 81 P BEY, C C YANG
UNCLASSIFIED NRL-MR-4669 NL

1 OF 1
AD A
108364

END
DATE
FILMED
01-82
DTIC


1.0
2.8
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

AD A1[illegible]364

(9) Ma... 1982

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) NRL-MR- NRL Memorandum Report 4669	2. GOVT ACCESSION NO. AD-A108 364	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) ON DIGITAL PROCESSING VIA PDQ FACTORIZATION		5. TYPE OF REPORT & PERIOD COVERED Preliminary report on a continuing NRL problem.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) P. Bey and C. C. Yang (11) 30 Nov 81		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, DC 20375 (12) 14		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NRL Problem 79-0719-0-2 Program Element 61153N14 RR0140241
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE November 30, 1981
		13. NUMBER OF PAGES 13
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (16) RR02402 (17) RR0240241		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital images
Pictorial database
Matrices
PDQ factorization
Rank
Approximation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this preliminary report, we present a computer program called MAT for implementing the PDQ factorization for digitized images of large pictorial data base. Within a mini-computer particular emphasis has been placed on the study of the efficiency of the algorithm in the storage and process of the images.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

251950

CONTENTS

ON DIGITAL PROCESSING VIA PDQ FACTORIZATION

I. Introduction

This report discusses a method of compressing data bases that describe large digital images. The ultimate objective of such a procedure is to effect savings in computer memory required to process images and to decrease the amount of data necessary to transmit images over a communication channel.

A digital picture (of photograph or video image) can be represented as a matrix consisting of rectangular blocks of size m x n where m and n are the number of pels (picture elements) in the horizontal and vertical directions, respectively. To process such a picture most effectively (in terms of processing time and quantities of the pictures) by a computer, it is desirable that the main memory of the computer, during the computation, has enough capacity for storage of the entries of the whole image matrix. Unfortunately, this is not the case for most modern minicomputers. Therefore, different adaptive techniques have been devised to cope with this situation. For instance, C. B. Moler and G. W. Stewart [2] described a matrix factorization, the so called PDQ factorization technique and applied it to image representation. The factorization is closely related to SVD (singular value decomposition technique developed by H. C. Andrews and C. L. Patterson [1]). The merit of PDQ factorization over SVD is that it requires, during computation, much less arithmetic and the entire maxtrix need not be stored in the main memory.

We have applied the PDQ factorization technique to various digitized pictures, and studied their approximations by different ranks with fixed initial column vector and observed the relationships between initial column vectors P_1 and resolution of the approximated image with fixed rank.

In the implementation of PDQ factorization, we have developed a computer program called MATRXVAX written in Fortran. This program is described in some detail.

II. Analysis of PDQ Factorization

Specifically, the PDQ factorization aims to express a matrix A with m rows and n columns as

$$A = PDQ^T, \qquad \text{(E)}$$

Manuscript submitted September 23, 1981.

where P is an m x m orthogonal matrix, Q^T is the transpose of Q, and D is a lower bidiagonal matrix, i.e. whose only nonzero entries are

$$_{i} = d_{i,i}, \quad i = 1,2,\text{---}\min(m,n) \; ,$$

$$_{i} = d_{i,i-1}, \; i = 2 \;,\text{---}\min(m+1,n) \; .$$

Assume $m \leq n$,

$$D = \begin{pmatrix} d_{11} & 0 & & & & \\ d_{12} & d_{22} & & & & \\ 0 & d_{23} & d_{33} & & & \\ \ddots & \ddots & \ddots & & & \\ & & d_{m,m-1} & d_{m,m} & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & d_{m,n} \end{pmatrix}$$

In our computer program, each column vector of P and Q is a normalized one. Given a (nonzero) initial column (or column vector) P_1 from A and, one can construct matrices P, D, and Q in (E) by some recursive formulae. The important feature of these recursive formulae is that, during the computations, the columns a_1, a_2, --- a_n of A are being referenced sequentially. We also note that, in practice, the inexact arithmetic due to the roundoff error may disturb the orthogonality as well as the normalization. Note that the choice of the initial column P_1 affects the whole structure of P, Q and D. An approximation A_r of A by specific P, Q and D is formed as follows:

$$A_r = P_r \, D_r \, Q_r^{\,T},$$

where P_r, Q_r are the matrices formed from the first r columns of P and Q (in (E)), and D_r is the r x r lower bidiagonal matrix with α_i, i = 1,2,...,r on the diagonal and β_i, i = 2, ..., r on the subdiagonal. Clearly A_r is a reduced matrix (of A) of rank $\leq$ r.

Moler and Stewart [2] showed, using theoretical studies of the Lanczos algorithm, that A_r will be a fairly accurate approximation to A even for quite small values of r. If one compares the storage for A and A_r, their ratio will be

$$R(m, n, r) = \frac{mr+2r+nr}{mr} .$$

Particularly, when $m = n$, we have

$$R(m, n, r) = \frac{2nr+2r}{n^2}$$

clearly in order to save storage it is necessarily that $2nr < n^2$.

That is

$$2r < n.$$

Thus PDQ decomposition will provide significant saving of storage and/or transmitting time if r is chosen much smaller than n.

It seems that, in general, for a matrix of low rank (its corresponding image is of relatively low detail), the PDQ factorization will be effective in processing and restoring the picture. In practice, it is not clear how to choose the value of r in order to have an efficient representation of a picture A_n with less computer storage and processing time. However, from our experiments, so far, $\frac{n}{4}$ seems to be a suitable threshold for the reduced rank r. Of course, the properties that lead to such low numerical rank remain to be studied.

Recall that [1] Andrews defines a condition number $C_R(G) = d_1/d_K$ in the discussion of the potentially efficient representation of the image (picture) in terms of its eigenimages, where G is the original picture (matrix), d_1 is the maximal eigenvalue, and d_k the minimal eigenvalue ($\neq 0$) in the SVD decomposition for G.

Similarly, we would like to define in PDQ factorization a condition number for reduced matrix of rank k: $C_k(A) = \mathrm{Max}\{d_i, g_i, i = 1, 2, \ldots k\}$,

where $d_i = \max\{\alpha_1, \alpha_2, \ldots \alpha_i\}/\min\{(\alpha_1, \alpha_2, \ldots, \alpha_i)\}$

and $g_i = \max\{\beta_1, \beta_2, \ldots \beta_i\}/\min\{(\beta_1, \beta_2, \ldots, \beta_i)\}$

A threshold for r values would be a value of k such that both $C_k(A)$ and $C_{k+1}(A)$ as well as their difference are large (of course, here the quantity of largeness remains to be decided). Another way that one may measure the closeness between A and A_r is by defining the quantity

$$D(A, A_r) = 1 - \frac{\sum_{i=1}^{r} |\alpha_i| + \sum_{j=1}^{r-1} |\beta_j|}{\sum_{i=1}^{n} |\alpha_i| + \sum_{j=1}^{n-1} |\beta_j|}$$

Clearly if $r = n$, then $D(A, A_r) = 0$. Therefore a threshold that one may consider for the approximated value r is to require $D(A, A_r)$ to be sufficiently small. We hope to extend our efforts to include the study of the threshold of r values in our next report.

III. Description of the Matrix Factorization Software.

Because of the limited memory of the NOVA 800, our principal computing tool, factorization of the image matrix is accomplished by the program MATRXVAX running on the VAX 11/780. A formatted image disk file 'MATAF' is created by the program VAXDATA on the NOVA 800 and written onto a magnetic tape by the program MAGTAP using a blocking factor equal to 1. Program MATRXVAX (1) reads the image magnetic tape file into memory on the VAX, (2) factors the image matrix into the matrices P,D and Q^T of specified rank r (3) reconstruct the image PDQ^T onto a magnetic file and (4) writes the matrices P, D and Q^T onto a magnetic tape file.

The image and matrix magnetic tape files created by VAX are written into disk files 'MATAFIN' and 'MATP', respectively, by the Program MAGTAP on the NOVA 800. The program WRTMATAFIN writes the reconstructed image stored in 'MATAFIN' on the COMTAL CRT and the program XCMT writes the COMTAL disk file onto magnetic tape. Images of rank r_i, where $r_i \leq r$ are written either on the COMTAL CRT by the program XMTCT or Optronics C-4300 Colorwriter by the Image Writing Software System PWREC.

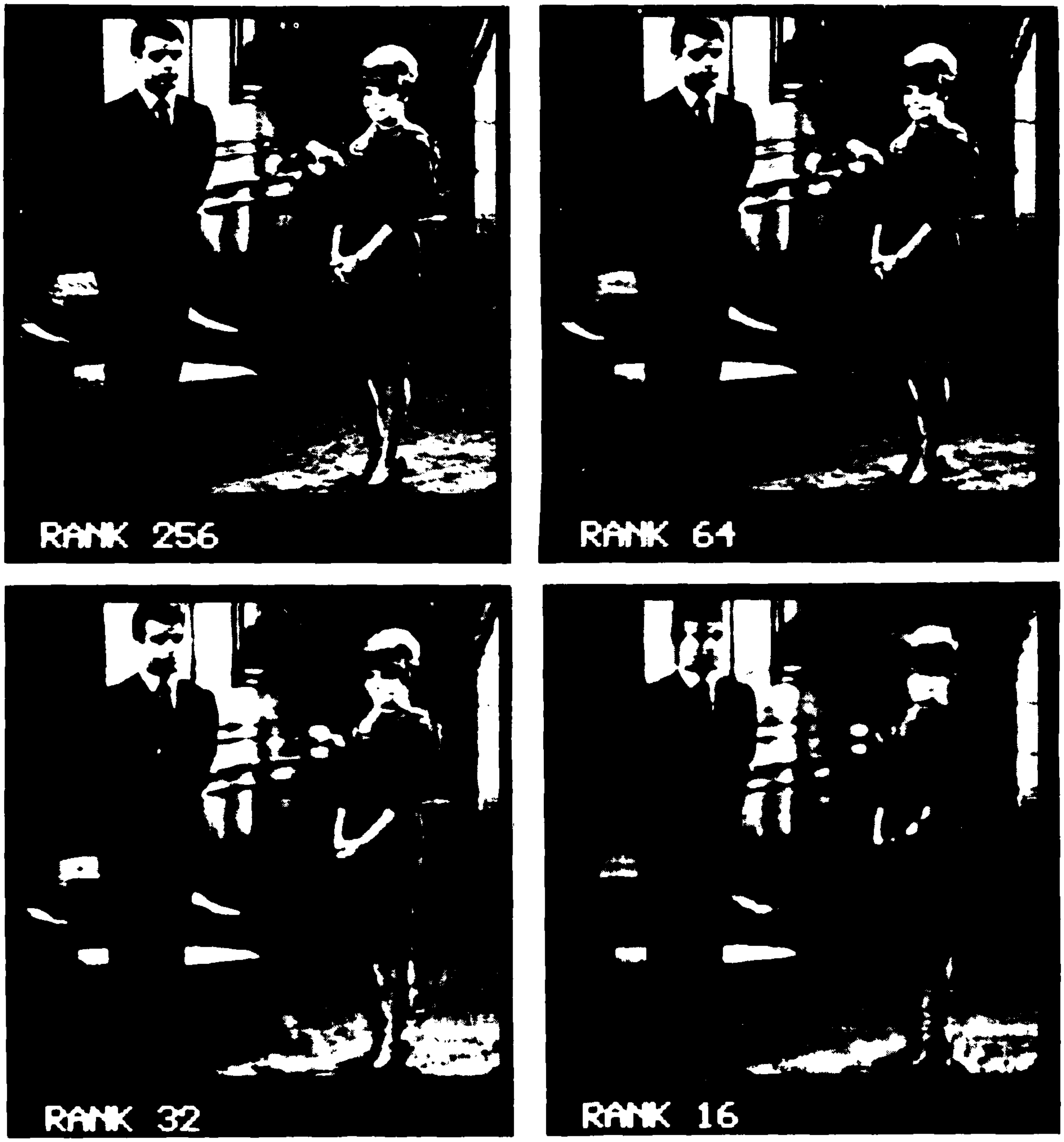
IV. Test Results (Pictures)

V. PDQ Factorization for Blurred Images

Denote the original image by A (without loss of generality, we may assume A is a square matrix). Suppose that the noise is modelled by the presence of two unknown matrices N_1 and N_2 such that for any given image A, $B = N_1A+N_2$ will be the blurred image of A. In order to apply PDQ factorization to A, we first have to determine the two unknown matrices. To do this we use identity matrices I and 2I to define

$$B_1 = N_1I + N_2 \text{ and } B_2 = 2N_1I + N_2.$$

Then B_1 and B_2 are the two known matrices. Therefore $N_1 = B_2 - B_1$ and $N_2 = 2B_1 - B_2$. Once N_1 and N_2 are obtained and stored, then for each blurred image B, we apply $N_1^{-1}(B-N_2)$ (provided that N_1^{-1} exists). We get the original image A and then proceed with PDQ factorization.


RANK 256
RANK 64
RANK 32
RANK 16

VI. Concluding Remarks

The above results will be used for further exploration of the PDQ factorization to digital images. As part of this work efficiency of factorization needs to be improved.

So far, we have seen, through some tests that the PDQ factorization did provide significant saving of storage as well as processing time in the representation of some ordinary pictures. We hope that further research will shed some light on the underlying theory. Our next topics of studies will be (i) properties of images (matrices) whose PDQ factorizations produce good (effective) approximations (i.e., the effective numerically reduced rank r is much less than the rank n of the original matrix, say, rank r less than n/4). (ii) Applying PDQ factorization to the enhancement of the images.

VII. List of Programs

```
                              ******************************************
                                               VAXDATA

                              08/28/81                      10:35:20
                              CURRENT DIR, YANG

                              ******************************************

C      RDOS FILE NAME:VAXDATA
C      INPUT DATA TO VAX FROM TAPE
       PARAMETER N=128,M=256
       DIMENSION IA(N),P(M),IB(M)
       COMMON/PPAP/A,B
       DATA A,B/0.,1./
       PF(X)=A*X+B
       ACCEPT 'MATRIX DIMENSION = ',NA
       ACCEPT 'MATRIX RANK = ',NR
       CALL OPEN(2,'MATAF',2,IER)
       WRITE(2,10)NA,NR
10     FORMAT(1X,2(I3,1X))
       ACCEPT 'TAPE FILE NUMBER = ',NFILE
       I2=NA/8
       ICNT=0
       IF(NFILE.EQ.0)GO TO 100
       CALL SKFLMT(IER,NFILE)
100    ILL=0
       CALL READMT(IER,IA,ILL)
       IF(IER.NE.5)GO TO 120
110    DO 130 J=1,NA
       IB(J)=0
130    CONTINUE
       GO TO 125
120    CALL FMLN(IA,IB,NA/2)
125    DO 20 I=1,I2
       K1=(I-1)*8+1
       K2=K1+7
       WRITE(2,30)(IB(L),L=K1,K2)
30     FORMAT(1X,8(I6,2X))
20     CONTINUE
       ICNT=ICNT+1
       IF(ICNT.EQ.NA)GO TO 160
       IF(IER.EQ.5)GO TO 110
       GO TO 100
160    CONTINUE
       DO 50 I=1,NA
       P(I)=PF(FLOAT(I))
```

```
50    CONTINUE
      DO 60 I=1,I2
      K1=(I-1)*8+1
      K2=K1+7
      WRITE(2,70)(P(L),L=K1,K2)
70    FORMAT(1X,8(F6.0,1X))
60    CONTINUE
      CALL CLOSE(2,IER)
      STOP
      END
```

****
MATRXVAX

08/28/81 10:47:46
CURRENT DIR. YANG

****

```
C     RDOS FILE NAME: MATRXVAX
C     MATRIX FACTORIZATION
      PARAMETER N=10,NRR=10
      COMMON/MATR/ Q(N,NRR),P(N,NRR),PT(N),QT(N),PTI(N),QTI(N),
     1ALPHA(N),BETA(N),IAR(N,N),IA(N,N)
X     OPEN(UNIT=21,TYPE='OLD',FILE='MATA.DAT')
X     OPEN(UNIT=23,TYPE='NEW',FILE='MATAF.DAT')
X     OPEN(UNIT=24,TYPE='NEW',FILE='MATP.DAT')
      READ(21,460)NA,NR
460   FORMAT(2(I3,1X))
      DO 420 J=1,256
      DO 410 I=1,32
      K1=(I-1)*8+1
      K2=K1+7
      READ(21,430)(IA(L,J),L=K1,K2)
430   FORMAT(8(I6,2X))
410   CONTINUE
420   CONTINUE
      DO 440 I=1,32
      K1=(I-1)*8+1
      K2=K1+7
      READ(21,450)(P(L,1),L=K1,K2)
450   FORMAT(8(F6.0,1X))
440   CONTINUE
      BETA(1)=0.
      SUM=0.
      DO 2 I=1,NA
      SUM=SUM+P(I,1)*P(I,1)
2     CONTINUE
      SUM=SQRT(SUM)
      DO 4 I=1,NA
      P(I,1)=P(I,1)/SUM
4     CONTINUE
      DO 20 I=1,NA
      SUM=0.
      DO 10 K=1,NA
      SUM=SUM+FLOAT(IA(K,I))*P(K,1)
10    CONTINUE
      Q(I,1)=SUM
20    CONTINUE
      SUM=0.
      DO 30 I=1,NA
      SUM=SUM+Q(I,1)*Q(I,1)
30    CONTINUE
      ALPHA(1)=SQRT(SUM)
      DO 40 I=1,NA
      Q(I,1)=Q(I,1)/ALPHA(1)
```

```
40    CONTINUE
      J=1
50    J=J+1
      IF(J.GT.NR)GO TO 230
      DO 70 J2=1,NA
      SUM=0.
      DO 60 K=1,NA
      SUM=SUM+FLOAT(IA(J2,K))*Q(K,J-1)
60    CONTINUE
      PT(J2)=SUM-ALPHA(J-1)*P(J2,J-1)
70    CONTINUE
      K2=J-1
      DO 100 J2=1,NA
      SUM1=0.
      DO 90 K=1,K2
      SUM=0.
      DO 80 JJ=1,NA
      SUM=SUM+PT(JJ)*P(JJ,K)
80    CONTINUE
      SUM1=SUM1+SUM*P(J2,K)
90    CONTINUE
      PTI(J2)=PT(J2)-SUM1
100   CONTINUE
      SUM=0.
      DO 110 I=1,NA
      SUM=SUM+PTI(I)*PTI(I)
110   CONTINUE
      BETA(J)=SQRT(SUM)
      DO 120 I=1,NA
      P(I,J)=PTI(I)/BETA(J)
120   CONTINUE
      DO 170 J2=1,NA
      SUM=0.
      DO 160 K=1,NA
      SUM=SUM+FLOAT(IA(K,J2))*P(K,J)
160   CONTINUE
      QT(J2)=SUM-BETA(J)*Q(J2,J-1)
170   CONTINUE
      K2=J-1
      DO 200 J2=1,NA
      SUM1=0.
      DO 190 K=1,K2
      SUM=0.
      DO 180 JJ=1,NA
      SUM=SUM+QT(JJ)*Q(JJ,K)
180   CONTINUE
      SUM1=SUM1+SUM*Q(J2,K)
190   CONTINUE
      QTI(J2)=QT(J2)-SUM1
200   CONTINUE
      DO 210 I=1,NA
      SUM=SUM+QTI(I)*QTI(I)
210   CONTINUE
      ALPHA(J)=SQRT(SUM)
      DO 220 I=1,NA
      Q(I,J)=QTI(I)/ALPHA(J)
220   CONTINUE
      GO TO 50
230   CONTINUE
      DO 330 K=1,NA
      DO 320 I=1,NA
      SUM=0.
      DO 310 J=1,NR
      SUM=SUM+P(I,J)*ALPHA(J)*Q(K,J)
      IF(J.EQ.1)GO TO 310
      SUM=SUM+P(I,J)*BETA(J)*Q(K,J-1)
310   CONTINUE
      IAR(I,K)=IFIX(SUM)
320   CONTINUE
      DO 330 J1=1,256
```

```
      DO 330 I1=1,32
      K1=(I1-1)*8+1
      K2=K1+7
      WRITE(23,431)(IAR(L,J1),L=K1,K2)
431   FORMAT(1X,8(I6,2X))
330   CONTINUE
X     ACCEPT 340,JYN
      READ(11,340)JYN
340   FORMAT('CONTINUE? (1=YES,0=NO) ',I3)
      IF(JYN.EQ.0)GO TO 360
X     ACCEPT 350,NR1
      READ(11,350)NR1
350   FORMAT('MATRIX RANK = ',I3)
360   CONTINUE
      I2=NR/4
      DO 370 I=1,I2
      K1=(I-1)*4+1
      K2=K1+3
      WRITE(24,380)(ALPHA(L),L=K1,K2)
      WRITE(24,380)(BETA(LL),LL=K1,K2)
380   FORMAT(1X,4(E12.6,1X))
370   CONTINUE
      NR2=NR/4
      DO 400 J=1,256
      DO 400 I=1,NR2
      K1=(I-1)*4+1
      K2=K1+3
      WRITE(24,380)(P(J,L),L=K1,K2)
      WRITE(24,380)(Q(J,LL),LL=K1,K2)
400   CONTINUE
      IF(JYN.EQ.0)GO TO 390
      NR=NR1
      GO TO 230
390   CONTINUE
X     CLOSE(UNIT=21,DISPOSE='DELETE')
X     CLOSE(UNIT=23,DISPOSE='SAVE')
X     CLOSE(UNIT=24,DISPOSE='SAVE')
      STOP
      END
```

```
                                        ****************************************
                                                               WRTMATAFIN

                                        08/28/81                      10:48:17
                                        CURRENT DIR,  YANG

                                        ****************************************
```

```
C     RDOS FILE NAME: WRTMATAFIN
C     WRITES IMAGE FROM VAX
C     NOTE: WHEN FORMAT WRITTEN WITH (1X,    ) TO VAX
C     WORD MUST BE READ WITH NOVA USING (1X,  ) FORMAT
C     WORD MUST BE READ WITH VAX USING (      ) FORMAT
C     I.E. 1X, DELETED FROM FORMAT STATEMENT
      PARAMETER N=512,M=256
      DIMENSION IA(N),IB(M)
      CALL OPEN(1,'MATAFIN',2,IER)
      CALL OPENCT(IER)
      ACCEPT 'TO IMAGE # = ',IOUT
      ACCEPT 'DISPLAY CODE = ',ICODE
      ACCEPT 'MATRIX DIMENSION = ',NA
      CALL DSPL(IER,ICODE)
      DO 20 J=1,256
      DO 10 I=1,32
      K1=(I-1)*8+1
      K2=K1+7
```

```
      READ(1,30)(IA(L),L=K1,K2)
X     WRITE(10,30)(IA(LL),LL=K1,K2)
30    FORMAT(1X,8(I6,2X))
      DO 40 M=K1,K2
      IF(IA(M).LT.0)IA(M)=0
      IF(IA(M).GT.255)IA(M)=255
40    CONTINUE
10    CONTINUE
      CALL PACK(IA,IB,NA)
      CALL WTIMAGE(IER,IOUT,J,IB,-NA/2)
20    CONTINUE
      CALL CLOSE(1,IER)
      STOP
      END
```

```
****************************************
                MATPQTP

08/28/81                    10:48:30
CURRENT DIR,  YANG

****************************************
```

```
C      NOUS FILE NAME:MATPQTP
C     IMAGE CONVERSION BY MULTIPLICATION OF P,Q AND D
C     MATRIX RANK NR0 RUN ON VAX MUST BE MULTIPLE OF 4
      PARAMETER N=1024,M=2048,NN=256,NM=128
      DIMENSION P(N),Q(N),ALPHA(NN),BETA(NN),IA(NN),IB(NM)
      COMMON/INTS1/IP(M)
      COMMON/INTS2/IQ(M)
      EQUIVALENCE (IP(1),P(1)),(IQ(1),Q(1))
      ACCEPT 'NEW RUN? (1=YES,0=NO) = ',JYN
      ACCEPT 'MAX MATRIX RANK NR0 = ',NR0
      ACCEPT 'MATRIX RANK = ',NR
      ACCEPT 'TO MT0:',IFL
      CALL OPEN(3,'MATP',2,IER)
      NLINES=1024/NR0
      NTRANS=NR0/4
      NWDS=1024
      NBLKS=8
      I2=NR0/4
      DO 30 I=1,I2
      K1=(I-1)*4+1
      K2=K1+3
      READ(3,20)(ALPHA(L),L=K1,K2)
      READ(3,20)(BETA(L),L=K1,K2)
20    FORMAT(1X,4(E13.6,1X))
30    CONTINUE
      IF(JYN.EQ.0)GO TO 32
      CALL CFILW('PMAT',3,200,IER)
      IF(IER.NE.1)TYPE 'FILE ERROR'
      CALL CFILW('QMAT',3,200,IER)
      IF(IER.NE.1)TYPE 'FILE ERROR 2'
32    CALL OPEN(1,'PMAT',2,IER)
      CALL OPEN(2,'QMAT',2,IER)
      IF(JYN.EQ.0)GO TO 52
      IMAX=NWDS/4
      DO 50 J=1,NTRANS
      DO 40 I=1,IMAX
      K1=(I-1)*4+1
      K2=K1+3
      READ(3,20)(P(L),L=K1,K2)
      READ(3,20)(Q(LL),LL=K1,K2)
40    CONTINUE
      CALL WRBLK(1,(J-1)*NBLKS,IP,NBLKS,IER)
      CALL WRBLK(2,(J-1)*NBLKS,IQ,NBLKS,IER)
X     TYPE 'IP(I),I=1,32 ',(IP(IXX),IXX=1,32)
X     TYPE 'IQ(I),I=1,32',(IQ(IXXX),IXXX=1,32)
```

```
50      CONTINUE
52      CONTINUE
        CALL RLSE('MT0',IER)
        CALL INIT('MT0',IER)
        IF(IFL.EQ.0)GO TO 200
        CALL SKFLMT(IER,IFL)
200     CONTINUE
        DO 160 K=1,NTRANS
        CALL RDBLK(2,(K-1)*NBLKS,IQ,NBLKS,IER)
        DO 160 K1=1,NLINES
        KJ=(K1-1)*NKR
        II=0
        DO 150 I=1,NTRANS
        CALL RDBLK(1,(I-1)*NBLKS,IP,NBLKS,IER)
        DO 150 I1=1,NLINES
        IJ=(I1-1)*NR0
        SUM=0.
        DO 140 J=1,NK
        SUM=SUM+P(IJ+J)*ALPHA(J)*U(KJ+J)
        IF(J.EQ.1)GO TO 140
        SUM=SUM+P(IJ+J)*BETA(J)*Q(KJ+J-1)
140     CONTINUE
        II=II+1
        IA(II)=IFIX(SUM)
        IF(IA(II).LT.0)IA(II)=0
        IF(IA(II).GT.255)IA(II)=255
150     CONTINUE
        CALL PACK(IA,IB,256)
        CALL WRITMT(IER,IB,-128)
160     CONTINUE
        CALL MTENDF(IER)
        CALL CLOSE(1,IER)
        CALL CLOSE(2,IER)
        CALL CLOSE(3,IER)
        CALL RLSE('MT0',IER)
        STOP
        END
```


Acknowledgement

We would like to thank Dr. Warren W. Willman for his assistance in running the program MATXVAX on VAX 11/780. Also thanks to Dr. Igor Jurkevich for his helpful suggestions as to the form of this report.

References

[1] H. C. Andrews and C. L. Patterson, Outer product expansion and their uses in Digital Image Processing, Amer. Math. Monthly, Vol. 1, No. 82, June 1975, pp. 1-13.

[2] C. B. Moler & G. W. Stewart, An Efficient Matrix Factorization for Digital Image Processing, LASL Tech. Report, LA-7637-MS, 1979.

01